

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BIGDATA AND CLOUD COMPUTING LAB**

**(B20EF0605)**

**for**

**Sixth Semester**

**B.Tech in Computer Science and Engineering**

|  |  |
| --- | --- |
| **Name** |  |
| **SRN** |  |
| **Branch** |  |
| **Semester** |  |
| **Section** |  |
| **Academic Year** |  |

### Vision of the Department

### Department of Computer Science and Engineering aspires to create a pool of high-calibre technologists and researchers in the field of computer science and engineering who have potential to contribute for development of the nation and society with their expertise, skills, innovative problem-solving abilities and strong ethical values.

### Mission of the Department

MD1: To create a center of excellence where new ideas flourish and from which emerge tomorrow’s researchers, scholars, leaders, and innovators.

MD2: Provide quality education in both theoretical and applied foundations of computer science and related inter-disciplinary areas and to train students to effectively apply the education to solve real-world problems.

MD3: Amplify student’s potential for life-long high-quality careers and give them a competitive in the ever-changing and challenging global work environment.

MD4: Forge research and academic collaboration with industries and top global universities in order to provide students with greater opportunities.

MD5: Support the society by encouraging and participating in technology transfer.

**Course Outcomes (Cos):**

On successful completion of this course; student shall be able to:

|  |  |  |  |
| --- | --- | --- | --- |
| CO# | Course Outcomes | POs | PSOs |
| CO1 | Understand the fundamentals of Technologies. | 1,2,3,4,5,9,10,11 | 1,2,3 |
| CO2 | Demonstrate Big Data Processing with PySpark to solve simple real world problems.  . | 1,2,3,4,5,9,10,11 | 1,2,3 |
| CO3 | Illustrate the fundamentals of Machine Learning for Big Data Analytics | 1,2,3,4,5,9,10,11 | 1,2,3 |
| CO4 | Create Machine Learning pipelines for Big Data Applications | 1,2,3,4,5,9,10,11 | 1,2,3 |
| CO5 | Design Real-Time Analytics with PySpark for real world Applications. | 1,2,3,4,5,9,10,11 | 1,2,3 |
| CO6 | Develop data and processing models using Python based PySpark for real world Big data Applications | 1,2,3,4,5,9,10,11 | 1,2,3 |

**Course Articulation Matrix**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course  Outcomes | Program Outcomes | | | | | | | | | | | | | | |
| **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| CO1 | 3 | 2 | 3 | 3 | 3 |  |  |  | 1 | 1 | 1 | 1 | 3 | 3 | 3 |
| CO2 | 3 | 3 | 3 | 3 | 3 |  |  |  | 2 | 1 | 2 | 3 | 3 | 3 | 3 |
| CO3 | 3 | 3 | 3 | 3 | 3 |  |  |  | 2 | 1 | 2 | 3 | 3 | 3 | 3 |
| CO4 | 3 | 3 | 3 | 3 | 3 |  |  |  | 3 | 3 | 2 | 3 | 3 | 3 | 3 |
| CO5 | 3 | 3 | 3 | 3 | 3 |  |  |  | 2 | 1 | 2 | 3 | 3 | 3 | 3 |
| CO6 | 3 | 3 | 3 | 3 | 3 |  |  |  | 3 | 3 | 2 | 3 | 3 | 3 | 3 |

Where, 1 (Low), 2(Medium) and 3 (High) represents strength of correlation between CO and PO.

**BLOOM’S LEVELOF THECOURSE OUTCOMES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CO# | Bloom’s Level | | | | | |
| Remember  (L1) | Understand  (L2) | Apply  (L3) | Analyze  (L4) | Evaluate  (L5) | Create  (L6) |
| CO1 |  | **√** |  |  |  |  |
| CO2 |  |  | **√** |  |  |  |
| CO3 |  | **√** |  |  |  |  |
| CO4 |  |  | **√** |  |  |  |
| CO5 |  |  | **√** |  |  |  |
| CO6 |  |  | **√** |  |  |  |

|  |
| --- |
| **Learning Objectives of the Course:** |
| The objectives of this course are to:   1. Discuss the fundamentals of Cloud Computing and Virtualization. 2. Demonstrate Big Data Processing with Python and PySpark. 3. Describe the implementation of Real-Time Analytics with PySpark in real world Applications. 4. Illustrate the working of Machine Learning applications suitable to handle big data using PySpark. |

|  |  |
| --- | --- |
| **Sl. No.** | **Problem Statement** |
| 1 | Demonstrate on the Virtual Environment on hypervisor. a) Communication between the VM’s. b) The backup and restore mechanism. |
| 2 | Demonstrate the mechanism of cloning and create a switch with multiple networks having the different VM’s. |
| 3 | Perform the following operations:   1. Create a SparkContext object 2. Create an RDD from set of words ("scala", "java", "hadoop", "spark",    "akka", "spark vs hadoop",   “pyspark",   "pyspark and spark") using *parallelize()* function. 3. Find the total count of the words in the RDD 4. Filter out and print the strings containing the word “spark” from the RDD |
| 4 | Given the two RDDs:   1. x created from the ordered pairs: ("spark", 1) and ("hadoop", 4) 2. y created from the ordered pairs: ("spark", 2), ("hadoop", 5).   Perform the join operation on the RDDs created above, and print the resulting RDD. |
| 5 | Create an RDD of set of numbers and perform the sum of these numbers using an *accumulator()* function in Spark context. |
| 6 | Create an RDD from the existing file having CSV data, using *read()* and *load()* functions and display the top 5 rows of the data set. And also display the statistical results from the data frame (Note: It only works for numerical values). |
| 7 | Create an RDD from the external text file. Find the word count in the text file using various transformation and action functions in PySpark. |
| 8 | Given the following data.  Data = [("James","Sales","NY",90000,34,10000),  ("Michael","Sales","NV",86000,56,20000),  ("Robert","Sales","CA",81000,30,23000),  ("Maria","Finance","CA",90000,24,23000),  ("Raman","Finance","DE",99000,40,24000),  ("Scott","Finance","NY",83000,36,19000),  ("Jen","Finance","NY",79000,53,15000),  ("Jeff","Marketing","NV",80000,25,18000),  ("Kumar","Marketing","NJ",91000,50,21000)  ], with the following schema  schema = ["employee\_name","department","state","salary","age","bonus"]  Perform the following using the aforementioned data.   1. create an RDD from the above data using its schema 2. create the PySpark dataframe from the RDD created. 3. Using groupBy() function, display the salaries of the employees state-wise. 4. Display the state-wise salaries that are greater than 1 lakh 5. Display the state-wise salaries in descending order.   Part B |
|  | Mini project on pyspark |

**Evaluation Sheet**

Name of the Faculty In charge:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Program/ Experiment No** | **Title of the Program /Experiment** | **Date of Execution** | **Max Marks** | **Marks Obtained** | **Signature of the Faculty I/c** |
| 1 |  |  | 10 |  |  |
| 2 |  |  | 10 |  |  |
| 3 |  |  | 10 |  |  |
| 4 |  |  | 10 |  |  |
| 5 |  |  | 10 |  |  |
| Internal Assessment-1 (IA-1) Consolidated Marks | | | 50/5 |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Program/ Experiment No** | **Title of the Program /Experiment** | **Date of Execution** | | **Max Marks** | **Marks Obtained** | **Signature of the Faculty I/c** |
| 6 |  |  | | 10 |  |  |
| 7 |  |  | | 10 |  |  |
| 8 |  |  | | 10 |  |  |
| 9 |  |  | | 10 |  |  |
| Additional Session (1-5) |  |  | | 10 |  |  |
| Internal Assessment-2 (IA-2) Consolidated Marks | | | | 50/5 |  |  |
|  | | | IA | Max Marks | Marks Obtained | Signature of the Faculty |
| IA1 |  |  |  |
| IA2 |  |  |  |
|  | | | IA1+IA2 | 20 |  |  |

Signature of the Faculty-in-charge with date

**Semester End Lab Examination Evaluation Procedure 2022-23**

|  |  |  |
| --- | --- | --- |
| **Name of the Lab:** Cloud Computing Lab | | |
| **Question** | **Parameters to be considered** | **Marks distribution** |
| **1-10** | Write-up | **10** |
| Conduction & Results | **10** |
| Viva | **5** |
| Total | **25** |

**Note:**

1. **Lab course is conducted for a total of 50 Marks:**
2. 25 Marks –Continuous Evaluation
3. 25 Marks Semester End Examination
4. Passing Criterion:
5. 08/25 in Semester End Examination
6. 20/50 overall

|  |  |
| --- | --- |
| **Session 3** | |
| **1** | **Problem Statement:** |
| Create multiple VM’s on a single physical machine. Provide a communication between them using and without using vClient. | |
| **2** | **Student Learning Outcomes:** |
| To install and setup the Virtual Machine  To check the connectivity between the VM’s | |
| **3** | **Theoretical Description:** |
| **Virtual Machine** (**VM**): is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.  There are different kinds of virtual machines, each with different functions:   1. System virtual machines (also termed full virtualization VMs) provide a substitute for a real machine. They provide functionality needed to execute entire operating systems. A hypervisor uses native execution to share and manage hardware, allowing for multiple environments which are isolated from one another, yet exist on the same physical machine. Modern hypervisors use hardware-assisted virtualization, virtualization-specific hardware, primarily from the host CPUs. 2. Process virtual machines are designed to execute computer programs in a platform-independent environment. | |
| **4** | **Requirements** |
| The following list of requirements as a starting point. Like physical computers, the virtual machines running under VMware Workstation generally perform better if they have faster processors and more memory.   1. PC Hardware - Standard x86-compatible personal computer, 400 MHz or faster CPU minimum (500 MHz recommended), Multiprocessor systems supported, 64-bit processor support for AMD64 Opteron, Athlon 64 and Intel IA-32e CPU (including "Nocona"). 2. Memory - 128 MB minimum (256 MB recommended) 3. Disk Drives - IDE and SCSI hard drives supported, up to 950GB capacity, At least 1GB free disk space recommended for each guest operating system and the application software used with it; if you use a default setup, the actual disk space needs are approximately the same as those for installing and running the guest operating system and applications on a physical computer. 4. Local Area Networking (Optional) 5. Host Operating System | |
| **5** | **Procedure** |
| Access the ESXi through the vCleint,   1. Open vSphere Client and Enter the followings:    1. IP of the server (eg: 128.0.78.222)    2. Enter the User Name (User name of the server)    3. Enter the password (password of the server)      1. To upload to datastore, Open the summary tab from the interface      1. Right Click on datastore and browse datastore.      1. Click on upload icon, press upload file, browse the file and upload.     **To create a VM, follow the procedure**   1. Right click on the server > New Virtual Machine      1. Select Custom and click next.      1. Give a name to the virtual machine and click next.      1. Select storage from the list and click next.      1. Select the virtual machine version (Select the latest version) and click next.      1. Select the type of guest Operating System and click next.      1. Select the number of virtual cores (Virtual sockets).      1. Select the amount of RAM and click next.      1. Don’t change anything in the networks page and click next.      1. Select SCSI controller as LSI Logic parallel      1. Click on create a new disk.      1. Enter the disk capacity (Atleast 30GB), select Thin provisioning and click next.      1. Review your configuration and finish. 2. Now select the VM and click on Edit Virtual machine settings. 3. Click on CD/DVD drive 1, select datastore ISO file and locate the ISO on the datastore.      1. Turn on the virtual machine and proceed with normal installation of OS in to that VM.   Repeat the same to create one more VM with different name/IP on same ESXi, then later open them through different/same clients. Now PING between them using their IP address. As ping <ip of other vm>   1. Power on the VM on ESXi      1. Power on the another VM on same ESXi      1. The console window can be displayed for those VM’s separately.      1. Find the IP address of those VM’s using “ipconfig” command in command prompt      1. Check the connectivity for the communication between those VM’s using PING command as   “ping <ip address of other VM>” | |

|  |  |  |
| --- | --- | --- |
| **Session 4** | | |
| **1** | **Problem Statement:** | |
| Demonstrate the communication of VM’s on different physical devices using ESXI. | | |
| **2** | **Student Learning Outcomes:** | |
| To install and setup the Virtual Machine on same and different ESXi.  To check the connectivity between the VM’s | | |
| **3** | **Theoretical Description:** | |
| **Virtual Machine** (**VM**): is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.  There are different kinds of virtual machines, each with different functions:   1. System virtual machines (also termed full virtualization VMs) provide a substitute for a real machine. They provide functionality needed to execute entire operating systems. A hypervisor uses native execution to share and manage hardware, allowing for multiple environments which are isolated from one another, yet exist on the same physical machine. Modern hypervisors use hardware-assisted virtualization, virtualization-specific hardware, primarily from the host CPUs. 2. Process virtual machines are designed to execute computer programs in a platform-independent environment. | | |
| **4** | **Requirements** | |
| The following list of requirements as a starting point. Like physical computers, the virtual machines running under VMware Workstation generally perform better if they have faster processors and more memory.   1. PC Hardware - Standard x86-compatible personal computer, 400 MHz or faster CPU minimum (500 MHz recommended), Multiprocessor systems supported, 64-bit processor support for AMD64 Opteron, Athlon 64 and Intel IA-32e CPU (including "Nocona"). 2. Memory - 128 MB minimum (256 MB recommended) 3. Disk Drives - IDE and SCSI hard drives supported, up to 950GB capacity, At least 1GB free disk space recommended for each guest operating system and the application software used with it; if you use a default setup, the actual disk space needs are approximately the same as those for installing and running the guest operating system and applications on a physical computer. 4. Local Area Networking (Optional) 5. Host Operating System | | |
| **5** | **Procedure** | |
| 1. Open the already created VM’s on same ESXi server using the different client browser’s on different physical machines. Then now ping between those VM’s of same ESXi. : Already demonstrated in the previous session. 2. Provide the communication between the VM’s created with different ESXi servers. So, create a new/ use the already created VM’s with different ESXi’s. Access the VM’s on different ESXi’s using the different client browser’s on different physical machines. Then now ping between them by using their IP address. 3. Login to two ESXi’s using two different vClient terminal.      1. Switch on the VM’s on those different ESXi’s, 2. Find the IP address of them using ipconfig 3. Check the connectivity for the communication using PING command | | |
| **Session 5** | | |
| **1** | | **Problem Statement:** |
| Illustrate the cloning of VM’s. | | |
| **2** | | **Student Learning Outcomes:** |
| To configure and maintain the VM’s.  To configure a multiple VM’s on multiple ESXi’s | | |
| **3** | | **Theoretical Description:** |
| **Cloning:** Cloning a virtual machine creates a duplicate of the virtual machine with the same configuration and installed software as the original.  Optionally, you can customize the guest operating system of the clone to change the virtual machine name, network settings, and other properties. This prevents conflicts that can occur if a virtual machine and a clone with identical guest operating system settings are deployed simultaneously.  Installing a guest operating system and applications can be time consuming. With clones, you can make many copies of a virtual machine from a single installation and configuration process. Clones are useful when you must deploy many identical virtual machines to a group.  With clones you can conveniently make complete copies of a virtual machine, without browsing a host file system or worrying if you have located all the configuration files.  **Open Virtualization Format** (**OVF**) is an open standard for packaging and distributing virtual appliances or, more generally, software to be run in virtual machines.  The standard describes an "open, secure, portable, efficient and extensible format for the packaging and distribution of software to be run in virtual machines". The OVF standard is not tied to any particular hypervisor or instruction set architecture. The unit of packaging and distribution is a so-called *OVF Package* which may contain one or more *virtual systems* each of which can be deployed to a virtual machine. | | |
| **4** | | **Requirements** |
| 1. You must be connected in order to clone a virtual machine. 2. To customize the guest operating system of the virtual machine, check that your guest operating system meets the requirements for customization. See [Guest Operating System Customization Requirements](https://pubs.vmware.com/vsphere-4-esx-vcenter/topic/com.vmware.vsphere.vmadmin.doc_41/vsp_vm_guide/deploy_vms_from_templates_and_clones/r_guest_customization_requirements.html#reference_8942A4F9FD484D60B09D7DA178F168A0). 3. To use a customization specification, you must first create or import the customization specification. 4. To use a custom script to generate the host name or IP address for the new virtual machine, configure the script. See [Configure a Script to Generate Computer Names and IP Addresses During Guest Operating System Customization](https://pubs.vmware.com/vsphere-4-esx-vcenter/topic/com.vmware.vsphere.vmadmin.doc_41/vsp_vm_guide/deploy_vms_from_templates_and_clones/t_configure_script_hostname.html#task_6801414C876E427BAEB11FD46D53FE69) | | |
| **5** | | **Procedure** |
| There are two ways to be followed,   1. **Using OVF.** 2. **Download/Upload a Folder of VM**   **Method I:**   1. Go to Edit Settings of the VM      1. Change the CD/DCD drive from “datasore” to “Client Device”.      1. Make sure that the VM is turned OFF 2. Select the VM which you want to clone 3. Go to File -> Export -> Export OVF Template      1. Select the destination where you wanted to store the copy of it        1. For importing a copy to same /different ESXi, 2. Go to file -> Deploy OVF Template      1. Select the source location from where the OVF file is to be deployed. 2. Select the exact copy of VM to be deployed among many files. | | |
| **Method II:**  1. Select the ESXi, Right click on Datastore -> browse datastore.   1. Select the folder of VM to be downloaded to physical machine 2. Select the option of “download a file from this data store to local machine” 3. Select the storage location 4. Go to vClient of any ESXi, Datastore -> Upload folder      1. Once the folder of VM is uploaded successfully, then open the folder 2. Right click on the .vms file -> add to inventory 3. Provide the name for the new VM 4. Select the destination ESXi 5. Once it is installed on to inventory then power ON the VM 6. Open through the Console 7. Select the options Moved – if you want to make the new copy replacing the old   Copied – if you wanted to create new by retaining the old. | | |

|  |  |
| --- | --- |
| **Session 6** | |
| **1** | **Problem Statement:** |
| Illustrate the backup-restore scenario. | |
| **2** | **Student Learning Outcomes:** |
| To figure out the different ways of backup and restore the sessions of VM | |
| **3** | **Theoretical Description:** |
| **A VMware snapshot:** is a copy of the virtual machine's disk file (VMDK) at a given point in time. Snapshots provide a change log for the virtual disk and are used to restore a VM to a particular point in time when a failure or system error occurs. Snapshots alone do not provide backup.  When you take a snapshot, you capture the state of the virtual machine settings and the virtual disk. If you are taking a memory snapshot, you also capture the memory state of the virtual machine. These states are saved to files that reside with the virtual machine's base files.  A snapshot consists of files that are stored on a supported storage device. A Take Snapshot operation creates .vmdk, -delta.vmdk, .vmsd, and .vmsn files. By default, the first and all delta disks are stored with the base .vmdk file. The .vmsd and .vmsn files are stored in the virtual machine directory. | |
| **4** | **Requirements** |
| 1. If you are taking a memory snapshot of a virtual machine that has multiple disks in different disk modes, verify that the virtual machine is powered off. For example, if you have a special purpose configuration that requires you to use an independent disk, you must power off the virtual machine before taking a snapshot. 2. To capture the memory state of the virtual machine, verify that the virtual machine is powered on. 3. To quiesce the virtual machine files, verify that the virtual machine is powered on and that VMware Tools is installed 4. Required privilege: Virtual machine**.**State**.** Create snapshot on the virtual machine. | |
| **5** | **Procedure** |
| 1. The VM, to which the backup is required should be in ON 2. Right click on VM -> Snapshot - > Take Snapshot 3. Provide the name for snapshot it may also be based on the time of backup. 4. Then make changes/updates in the current VM 5. For retrieving / get back to initial state of VM, 6. Right click on VM -> Snapshot -> Revert to Current Snapshot 7. Accept for the current modification to be done in the VM      1. Get back to original state of VM 2. If you ever taken too many snapshots for the same VM, need to manage and revert to required state of VM      1. Right click on VM -> Snapshot -> Snapshot manager      1. You can see the multiple snapshots taken for the VM and also current state of VM      1. You can select any snapshot and revert to respective state of VM | |

|  |
| --- |
| **Experiment 3** |
| **Problem Statement:** |
| Perform the following operations:  a. Create a SparkContext object  b. Create an RDD from set of words ("scala", "java", "hadoop", "spark", "akka", "spark vs hadoop", “pyspark", "pyspark and spark") using parallelize() function.  c. Find the total count of the words in the RDD  d. Filter out and print the strings containing the word “spark” from the RDD |
| **Student Learning Outcomes:** |
| After the learning this code snippet, the learner will able to understand about the concept of RDD and find the total count of words and able to print the strings containing particular word from the RDD |
| **Theoretical Description:** |
| * RDD (Resilient Distributed Datasets) in Spark is a fundamental data structure that represents an immutable, fault-tolerant collection of data that can be processed in parallel across a cluster of machines. * RDDs can be created from various data sources, including Hadoop Distributed File System (HDFS), local file systems, and external databases, and can be transformed using various operations such as map, filter, and reduce to perform complex computations. * RDDs are the basic building blocks of Spark applications and provide a high-level API for performing distributed data processing. |
| **Requirements:** |
| **PySpark / Google Colabs with Python and Java Latest Version for JVM** |
| **Program:** |
| **Step 1 : First, We need to create a Spark Session**  **Step 2 : Next Create a SparkContext Object**  sc = SparkContext()  **Step 3 :** **Create a RDD called by using sparkcontext object with parallelize method**  rdd = sc.parallelize(["scala", "java", "hadoop", "spark", "akka", "spark vs hadoop",  “pyspark", "pyspark and spark"])  **Step 4 :** **Find the total count of words in the RDD using Count( ) method**  total\_count = rdd.count()  **Step 5 :** **This line of code creates a new RDD (Resilient Distributed Dataset) called**  spark\_words\_rdd **by filtering the original RDD** rdd**. The filter condition is defined**  **using a lambda function which takes in each element** x **of the RDD and checks if**  **the string "spark" is present in it. Only the elements that satisfy this condition**  **are included in the new RDD.**  spark\_words\_rdd = rdd.filter(lambda x: "spark" in x)  **Step 6 : The line** spark\_words = spark\_words\_rdd.collect() **in a PySpark program creates a**  **Python list** spark\_words **by collecting all the elements of the Spark RDD**  spark\_words\_rdd **to the driver node. This operation brings all the data from the**  **distributed nodes into memory on the driver node, which can be a bottleneck in**  **large-scale distributed processing.**  spark\_words = spark\_words\_rdd.collect()  **Step 7 :** **In the specific context of PySpark, if** spark\_words **was obtained by collecting all**  **the elements of an RDD to the driver node (e.g.,** spark\_words =  spark\_words\_rdd.collect()**), then printing it will show the contents of the RDD as**  **a Python list.**  print(spark\_words)  **Complete Code without explanation**  sc = SparkContext()  rdd = sc.parallelize(["scala", "java", "hadoop", "spark", "akka", "spark vs hadoop",  “pyspark", "pyspark and spark"])  total\_count = rdd.count()  spark\_words\_rdd = rdd.filter(lambda x: "spark" in x)  spark\_words = spark\_words\_rdd.collect()  print(spark\_words) |
| **Output** |
| The output of the code will be a list of strings containing the word "spark":  ['spark', 'spark vs hadoop'] |

|  |
| --- |
| **Experiment 4** |
| **Problem Statement:** |
| Given the two RDDs:  a. x created from the ordered pairs: ("spark", 1) and ("hadoop", 4)  b. y created from the ordered pairs: ("spark", 2), ("hadoop", 5).  Perform the join operation on the RDDs created above, and print the resulting RDD. |
| **Student Learning Outcomes:** |
| After the learning this code snippet, the learner will able to understand about the concept of RDD, How to use Join Operation in Spark context in RDD. |
| **Theoretical Description:** |
| * Spark: Apache Spark is an open source Distribution Processing System used for bigdata workloads * RDD (Resilient Distributed Datasets) in Spark is a fundamental data structure that represents an immutable, fault-tolerant collection of data that can be processed in parallel across a cluster of machines. * RDDs can be created from various data sources, including Hadoop Distributed File System (HDFS), local file systems, and external databases, and can be transformed using various operations such as map, filter, and reduce to perform complex computations. * RDDs are the basic building blocks of Spark applications and provide a high-level API for performing distributed data processing. |
| **Requirements:** |
| **Jupiter notebook/ Google Colabs with Python and Java Latest Version for JVM** |
| **Program:** |
| **# Install Spark**  pip install spark  **# Install Pyspark**  Pip install pyspark  **# Create Context and Session for Spark**  from pyspark.context import SparkContext  from pyspark.sql.session import SparkSession  sc=SparkContext.getOrCreate()  spark=SparkSession(sc)  **#Using parallelize method, create RDD**  rdd1=sc.parallelize([("spark", 1),("hadoop", 4)])  rdd2=sc.parallelize([("spark", 2),("hadoop", 5)])  **#Perforn Join Operation on the created RDDs**  rdd=sorted(rdd1.join(rdd2).collect())  **#Print the Result**  Print(rdd) |
| **Output** |
| [('hadoop', (4, 5)), ('spark', (1, 2))] |

|  |
| --- |
| **Experiment 5** |
| **Problem Statement:** |
| Create an RDD of set of numbers and perform the sum of these numbers using an accumulator() function in Spark context. |
| **Student Learning Outcomes:** |
| After the learning this code snippet, the learner will able to understand about the concept of RDD and find the Sum of the Numbers using accumulator() function  in Spark context in RDD. |
| **Theoretical Description:** |
| * RDD (Resilient Distributed Datasets) in Spark is a fundamental data structure that represents an immutable, fault-tolerant collection of data that can be processed in parallel across a cluster of machines. * RDDs can be created from various data sources, including Hadoop Distributed File System (HDFS), local file systems, and external databases, and can be transformed using various operations such as map, filter, and reduce to perform complex computations. * RDDs are the basic building blocks of Spark applications and provide a high-level API for performing distributed data processing. |
| **Requirements:** |
| **PySpark / Google Colabs with Python and Java Latest Version for JVM** |
| **Program:** |
| **from pyspark import SparkContext**  **# create a Spark context**  **sc = SparkContext()**  **# create an RDD of set of numbers**  **rdd = sc.parallelize([{1, 2, 3}, {4, 5, 6}, {7, 8, 9}])**  **# define an accumulator**  **acc = sc.accumulator(0)**  **# use the accumulator to sum the numbers in the RDD**  **def add\_to\_acc(x):**  **global acc**  **acc += sum(x)**  **rdd.foreach(add\_to\_acc)**  **# print the result**  **print("Sum of numbers in RDD: ", acc.value)**  **# stop the Spark context**  **sc.stop()** |
| **Output** |
| **Sum of numbers in RDD: 45** |

|  |
| --- |
| **Experiment 6** |
| **Problem Statement:** |
| Create an RDD from the existing file having CSV data, using *read()* and *load()* functions and display the top 5 rows of the data set. And also display the statistical results from the data frame (Note: It only works for numerical values). |
| **Student Learning Outcomes:** |
| After the learning this code snippet, the learner will able to understand about the concept of RDD and Creation of .CSV file and insert 10 rows and 3 cloumns of numerical value and uplaod in to Files in colab .Students will also know about the datframe creation. |
| **Theoretical Description:** |
| * RDD (Resilient Distributed Datasets) in Spark is a fundamental data structure that represents an immutable, fault-tolerant collection of data that can be processed in parallel across a cluster of machines. * RDDs can be created from various data sources, including Hadoop Distributed File System (HDFS), local file systems, and external databases, and can be transformed using various operations such as map, filter, and reduce to perform complex computations. * RDDs are the basic building blocks of Spark applications and provide a high-level API for performing distributed data processing. |
| **Requirements:** |
| **PySpark / Google Colabs with Python and Java Latest Version for JVM** |
| **Program:** |
| **Step1:**  **pip install spark**  **step2:**  **pip install pyspark**  **Step3:**  **Create one .CSV file and insert 10 rows and 3 cloumns of numerical value and uplaod in to Files in colab.**  **Step4: Code**  **from pyspark.sql import SparkSession**  **# create a SparkSession object**  **spark = SparkSession.builder.appName("CSV RDD").getOrCreate()**  **# create an RDD from a CSV file**  **rdd = spark.read.format("csv").option("header", "true").load("/Book1.csv").rdd**  **# display the top 5 rows of the RDD**  **print(rdd.take(5))**  **# convert the RDD to a DataFrame**  **df = rdd.toDF()**  **# display the statistical results**  **df.describe().show()** |
| **CSV File** |
|  |
| **Output** |
| **[Row(radio='99.1', Newspaper='22.1', Sales='44.1'), Row(radio='20.1', Newspaper='23.4', Sales='55.1'), Row(radio='33.1', Newspaper='26.4', Sales='66.1'), Row(radio='44.1', Newspaper='123.1', Sales='88.1'), Row(radio='88.1', Newspaper='20.1', Sales='12.1')]**  **+-------+-----------------+-----------------+-----------------+**  **|summary| radio| Newspaper| Sales|**  **+-------+-----------------+-----------------+-----------------+**  **| count| 8 | 8 | 8|**  **| mean| 48.35 | 39.325 | 39.35000000000001|**  **| stddev|35.30746582321001|34.75755827360226|28.49937342669634|**  **| min| 10 | 123.1 | 12.1|**  **| max| 99.1 | 44.1 | 88.1|**  **+-------+-----------------+-----------------+-----------------+** |

|  |
| --- |
| **Experiment 7** |
| **Problem Statement:** |
| Create an RDD from the external text file. Find the word count in the text file using various transformation and action functions in PySpark. |
| **Student Learning Outcomes:** |
| After the learning this code snippet, the learner will able to understand about the concept of RDD and Creation of Text. Also find the number of words in text file. |
| **Theoretical Description:** |
| * RDD (Resilient Distributed Datasets) in Spark is a fundamental data structure that represents an immutable, fault-tolerant collection of data that can be processed in parallel across a cluster of machines. * RDDs can be created from various data sources, including Hadoop Distributed File System (HDFS), local file systems, and external databases, and can be transformed using various operations such as map, filter, and reduce to perform complex computations. * RDDs are the basic building blocks of Spark applications and provide a high-level API for performing distributed data processing. |
| **Requirements:** |
| **PySpark / Google Colabs with Python and Java Latest Version for JVM** |
| **Program:** |
| Step1: Import the "files" module from the "google.colab" library.  **from google.colab import files**  Step2: Upload file from system  **uploaded = files.upload()**  Step3: To install PySpark, you can use pip, which is a package installer for Python.  **pip install pyspark**  Step4: To import the "SparkSession" class from the "pyspark.sql" module.  **from pyspark.sql import SparkSession**  Step5: This line of code creates a SparkSession object named **spark** with the name **word\_count** assigned to it.  **spark=SparkSession.builder. appName("word\_count"). getOrCreate()**  Step6: This line of code reads a text file named **txtt.txt** into a DataFrame using the **read** method of the **spark** object.  **text\_file = spark.read.text("txtt.txt")**  Step7: his line of code transforms the **text\_file** DataFrame into an RDD (Resilient Distributed Dataset) and then applies a **flatMap** transformation to it. The **flatMap** transformation takes a function as input and applies it to each element of the RDD, resulting in a new RDD with the flattened results.  **words = text\_file.rdd.flatMap(lambda line: line.value.split(" "))**  Step8: Count the number of words in text file  **word\_count = words.count()**  Step9: Display word count  **print("Number of words in text file: ", word\_count)** |
| **Output** |
|  |
| **Text file** |
|  |

|  |
| --- |
| **Experiment 8** |
| **Problem Statement:** |
| Given the following data.  Data = [("James","Sales","NY",90000,34,10000),  ("Michael","Sales","NV",86000,56,20000),  ("Robert","Sales","CA",81000,30,23000),  ("Maria","Finance","CA",90000,24,23000),  ("Raman","Finance","DE",99000,40,24000),  ("Scott","Finance","NY",83000,36,19000),  ("Jen","Finance","NY",79000,53,15000),  ("Jeff","Marketing","NV",80000,25,18000),  ("Kumar","Marketing","NJ",91000,50,21000)  ], with the following schema  schema = ["employee\_name","department","state","salary","age","bonus"]  Perform the following using the aforementioned data.   1. create an RDD from the above data using its schema 2. create the PySpark dataframe from the RDD created. 3. Using groupBy() function, display the salaries of the employees state-wise. 4. Display the state-wise salaries that are greater than 1 lakh 5. Display the state-wise salaries in descending order. |
| **Student Learning Outcomes:** |
| After the learning this code snippet, the learner will able to understand about the concept of RDD and creation of data frames and schema ,Usage of groupby(). |
| **Theoretical Description:** |
| * RDD (Resilient Distributed Datasets) in Spark is a fundamental data structure that represents an immutable, fault-tolerant collection of data that can be processed in parallel across a cluster of machines. * RDDs can be created from various data sources, including Hadoop Distributed File System (HDFS), local file systems, and external databases, and can be transformed using various operations such as map, filter, and reduce to perform complex computations. * RDDs are the basic building blocks of Spark applications and provide a high-level API for performing distributed data processing. |
| **Requirements:** |
| **PySpark / Google Colabs with Python and Java Latest Version for JVM** |
| **Program:** |
| **Step-1**  pip install pyspark  from pyspark.sql import SparkSession  from pyspark.sql.functions import col,sum,avg,max  **step-2**  **# To solve the problem, we need to import necessary PySpark libraries and create a SparkSession #**  spark = SparkSession.builder \  .appName('SparkByExamples.com') \  .getOrCreate()  **Step-3: Create an RDD from the above data using its schema**  **# To create an RDD, we first define the schema for our data, which is provided in the problem.**  Data = [("James","Sales","NY",90000,34,10000),  ("Michael","Sales","NV",86000,56,20000),  ("Robert","Sales","CA",81000,30,23000),  ("Maria","Finance","CA",90000,24,23000),  ("Raman","Finance","DE",99000,40,24000),  ("Scott","Finance","NY",83000,36,19000),  ("Jen","Finance","NY",79000,53,15000),  ("Jeff","Marketing","NV",80000,25,18000),  ("Kumar","Marketing","NJ",91000,50,21000)  ]  **# We can now create an RDD using the parallelize() method of SparkContext, and pass in our data and the schema we defined.**  schema = ["employee\_name","department","state","salary","age","bonus"]  **Step - 4:Create the PySpark dataframe from the RDD created.**  **# We can create a PySpark dataframe from our RDD using the createDataFrame() method of SparkSession. We pass in our RDD and the schema we defined to this method.**  **df = spark.createDataFrame(data=Data, schema = schema)**  **df.printSchema()**  **df.show(truncate=False)**  **Step-5 : Using groupBy() function, display the salaries of the employees state-wise.**  **# To display the salaries of the employees state-wise, we can use the groupBy() method of PySpark dataframe.**  **df.groupBy("state").sum("salary").show()**  **# This will group the dataframe by "state" column and calculate the sum of "salary" column for each group.**  **Step-6 Display the state-wise salaries that are greater than 1 lakh**  **# To display the state-wise salaries that are greater than 1 lakh, we can add a filter to the above query.**  df.groupBy("state").sum("salary").filter("sum(salary) > 100000").show()  **# This will filter the result and display only those states where the sum of salaries is greater than 1 lakh.**  **Step-7: Display the state-wise salaries in descending order.**  **# To display the state-wise salaries in descending order, we can add an orderBy() method to the above query.**  df.groupBy("state").sum("salary").orderBy("sum(salary)", ascending=False).show() |
| **Output : Creation of schema** |
| **C:\Users\S3LAB30\Downloads\8th op.PNG** |
| **Output : Group by Function** |
| **C:\Users\S3LAB30\Downloads\8th op1.PNG** |
| **Output : groupby() for ascending order** |
| **C:\Users\S3LAB30\Downloads\8th op2.PNG** |

**Recommended Course Resources:**

1. <http://hadoop.apache.org/>
2. <https://www.macalester.edu/~shoop/sc13/hadoop/html/hadoop/wc-detail.html>
3. <https://en.wikipedia.org/wiki/K-means_clustering>
4. <https://github.com/himank/K-Means>
5. <https://www.arduino.cc/>
6. <https://www.arduino.cc/en/Main/arduinoBoardUno>
7. <https://en.wikipedia.org/wiki/Arduino>
8. [www.arduino.org/](http://www.arduino.org/)
9. [*www.vmware.com*](http://www.vmware.com)
10. http://www.vmware.com/products.html